

Evolutionary Intelligent Agents for e-Commerce: Generic Preference Detection with Feature Analysis

Sheng-Uei Guan, Tai Kheng Chan and Fangming Zhu

Department of Electrical and Computer Engineering

National University of Singapore

10 Kent Ridge Crescent, Singapore 119260

eleguans@nus.edu.sg

ABSTRACT

Product recommendation and preference tracking systems have been adopted extensively in e-commerce businesses. However, the heterogeneity of product attributes results in undesired impediment for an efficient yet personalized e-commerce product brokering. Amid the assortment of product attributes, there are some intrinsic generic attributes having significant relation to a customer's generic preference. This paper proposes a novel approach in the detection of generic product attributes through feature analysis. The objective is to provide an insight to the understanding of customers' generic preference. Furthermore, a genetic algorithm is used to find the suitable feature weight set, hence reducing the rate of misclassification. A prototype has been implemented and the experimental results are promising.

KEYWORDS: Generic Preference, E-Commerce, Generic Attributes, Feature Analysis, Genetic Algorithm

1 INTRODUCTION

Intelligent software agents are software entities that carry out some set of operations on behalf of a user. They normally possess certain form of intelligence together with some representation of user's goals or desires [1] [2] [3]. These software agents are usually autonomous, reactive, intelligent and even social. Smart agents should be able to anticipate customer needs and proactively take an action. Artificial intelligence built into e-commerce agents can provide services such as intelligent product filtering and brokering to patrons at online stores. In addition, e-commerce agents should have the ability to understand customer's requirements and personalize products recommendation.

Nevertheless, productive use of online resources is hampered by the colossal amount of information. As the number of products retailing in e-commerce websites increases exponentially, customers will have some difficulty in searching for a desired item. At Amazon.com [4], there are millions of products available with tens of millions of customers browsing through the website. However, Amazon.com manages to succeed in the e-commerce retailing industry with a personalized and intelligent recommendation system. Inevitably, personalization will be an indispensable contrivance for an efficacious and thriving e-commerce industry. Information seekers or online customers will require an individualized, autonomous system that can learn a customer's specific preference and search for relevant information.

The purpose of our research is to exploit the capabilities of intelligent e-commerce agents that can filter and recommend suitable products for a customer in an effective

way. This paper focuses on the detection of useful generic product attributes that are unaccounted for *per se*, as we believe these generic attributes generally contain valuable information in constructing customer-oriented recommendation.

Most similar approaches in the past can only identify specific tangible product attributes of interest to customers. In fact, many recommendation systems are not dynamic and flexible enough to adapt to the customer's preference changes. Unlike those approaches, our approach aims to probe in-depth the underlying preference of a customer. By detecting generic attributes, the system is able to understand the generic preference of a customer.

Generic preference is useful as it characterizes the intrinsic liking of a customer. For example, knowing that a customer prefers a 2.2 GHz Intel-chipset notebook is superficial for true understanding. Instead, if the system is able to know that the customer's preference is a "high speed" notebook or any computer that has Intel chips, it is an added advantage. Therefore, a better smart system can go further than just recommending localized preferred products – it can recommend extra products with certain generic attributes which match with the customer's preference.

This paper proposes an approach for discovering generic product attributes. The key element in this approach is the feature analysis of attributes, where product attributes are evaluated according to their distinctive features. The features of each attribute embody unique properties, which represent the signature of the attribute inherently (refer to section 3.2). In addition, to corroborate the strength of the attribute's

signature, weights are bound to each feature to symbolize the significance. With the weighted-features, each product attribute possesses a distinctive coded signature, which is used in the classification of matching attributes. An adaptation of the k-Nearest Neighbor (k-NN) classification approach [5] and product attribute ontology [6] [7] are used to identify the generic classes. Furthermore, a genetic algorithm (GA) is used to find the suitable feature weight set and reduce the rate of misclassification [8] [9].

A prototype has been implemented in Java. The prototype contains a population of evolutionary intelligent agents gathering customer feedback, processing product information, and recommending suitable products to the customer. Some simulations were run and the results proved to be satisfactory. The prototype is able to detect generic attributes with low misclassification rate. The GA deployed in the system, together with the k-NN classification approach, can successfully find the suitable feature weights. It is also observed that the system is able to adapt to abrupt customer-preference changes.

The paper is organized as follows. We first introduce some background on product recommendation and preference tracking systems in section 2. The system design issues are discussed in section 3. The evaluation of the design and corresponding findings can be found in section 4. Section 5 concludes the paper.

2 BACKGROUND

A great deal of research has been done to improve the efficiency and usability of e-commerce. The common goal is to analyze and understand a customer's needs prior to recommending suitable products through product brokering services. The pervasive penetration of e-commerce activities has enticed many researchers towards improving e-commerce transactions. Above all, product brokering agents have been studied extensively [10] [11] [12] [13], in which these agents have the capabilities to provide customer-oriented recommendations intelligently.

User preference is an important concept in predicting customer behaviors and recommending preferred products in personalized systems. Preference is the concept to make relation between a person and a target item which contains several kinds of attributes. Formalized preference models include positive and negative preference [14]. Preferred items are known as positive preference, and non-preferred items are known as negative preference.

A lot of research has targeted tracking customer preference in order to provide more customized recommendations. In the paper by Guo *et al.* [15], agents operate on behalf of customers in e-commerce negotiations. The agents retrieve the required information about their customer's preference structures. In another research, Shibata *et al.* [11] proposed an approach in which autonomous agents can learn customer-preference by observing the customer's reaction to contents recommended by agents.

An intelligent preference tracking research done by Guan *et al.* [10] made use of evolutionary ontology-based product brokering agents targeting m-commerce applications. The GA was used to tune parameters for tracking customer preference. The same research group led by Guan, designed another recommending system [16] that consists of agents with the ability to capture customer preference with product attribute tracking. The proposed solution is able to detect both quantifiable and non-quantifiable intangible product attributes through repeated customer feedbacks. However, though the research is successful in detecting intangible attributes hidden in the products, it was not designed to detect generic attributes that will provide clues on the customer's generic preference.

Other recommendation systems [17] [18] used various filtering approaches to elicit the customer's preference. Shahabi and Chen [17] employed a hybrid approach that combines collaborative filtering (CF) and content-based querying. The downside is that the system relied too heavily on customer profiles for providing accurate recommendation lists.

The common approach to handle product attributes is to assemble these attributes and assign weights with relevance to their importance to the customer. The weights are adjusted to reflect the customer's preference. Guan *et al.* [10] captured customer preference by requesting the customer to select the best product from a short list of products before adjusting the weights according to the feedback. A similar approach [11] also assigned weights to attributes, and these attributes weights were adjusted

through reinforcement learning. Though weights were being used to improve the robustness of the system, they are not flexible enough for true optimization.

A consumer's product preference is often influenced by product attributes that can vary from price to brand. Product ontology plays an important role in the discovery of user preference because it provides an initial guide in the direction where the preference might be heading [6] [19]. Many researchers have also made use of product ontology in the field of e-commerce to improve the understanding of customer preference. In this research, product ontology is one that categorizes and maps heterogeneous products taxonomically. In contrast, attribute ontology refers to one that categorizes and maps heterogeneous product attributes taxonomically. Figures 1 and 2 illustrate these two ontologies.

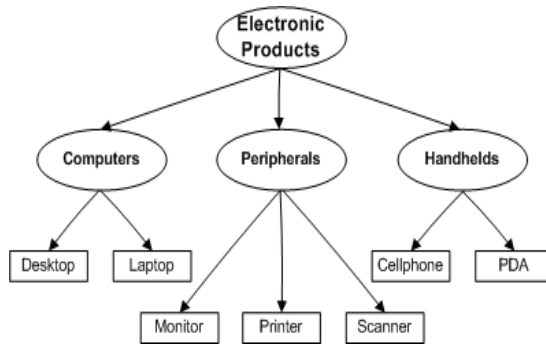


Figure 1: Product Ontology Example

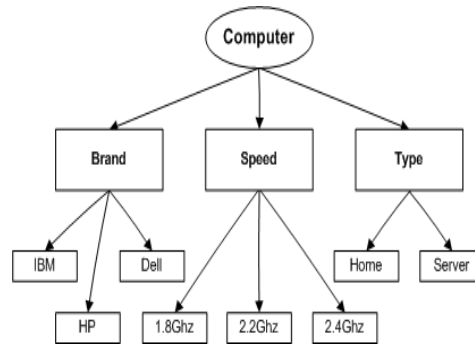


Figure 2: Product Attribute Ontology Example

It is observed from Figure 2 that attributes such as brand, speed and type are actually generic attributes of a computer product. These generic attributes are the super attributes of those specific attributes such as IBM, 1.8GHz and Server. Therefore, a

computer brand can either be IBM, HP or Dell. When we refer to a customer having some generic attribute preference, we are indicating that the customer gives more importance to the brand of a computer than speed or type. For example, if the user thinks that the brand of a computer is more important than the speed or type, the particular preference for brand will be useful for product brokering.

3 SYSTEM DESIGN

Figure 3 illustrates the system architecture. The user interface agent is embedded within the Java-based graphics user interface (GUI). Its main purpose is to assist the user in selections, feedbacks, retrieval of products, and presentation of recommendations.

The Database Centre maintains the product database, attribute ontology and user's preference in previous sessions. A population of agents is created by the system automatically, after which they undergo GA evolution at the Agent Training Center (refer to section 3.4). After a few generations of the GA, the results are presented to the user. Each agent possesses some form of knowledge, including feature set, its corresponding feature weights, attribute signature, and generic attributes or classes detected. Furthermore, each agent has a fitness level which is the decisive factor to the survival of the agent during evolution. The detailed design issues are presented in the following subsections.

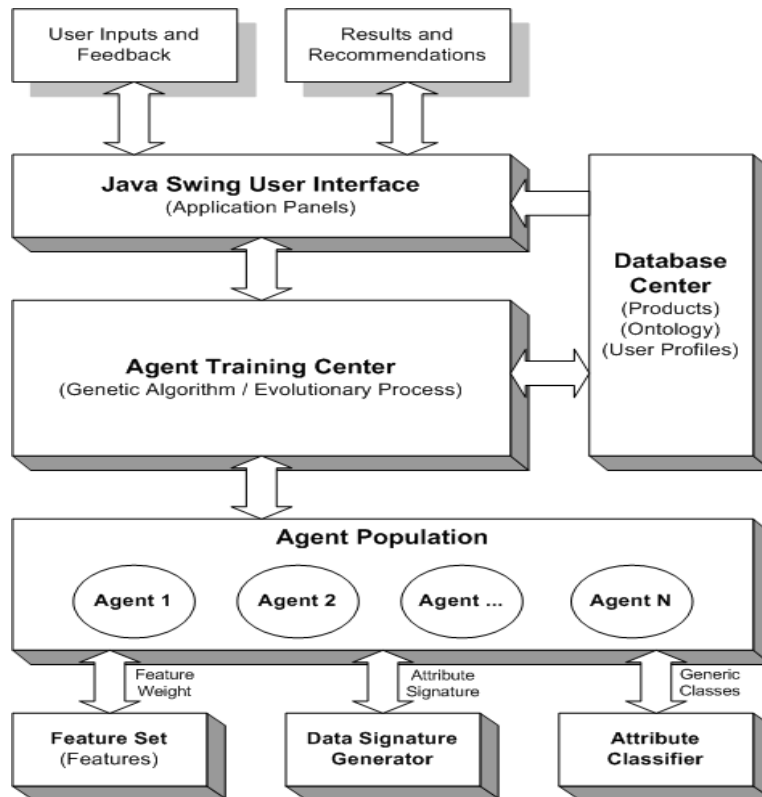


Figure 3: System Architecture

3.1 USER FEEDBACK AND RANKING SYSTEM

In order to understand the customer's preference, a series of feedbacks is used. The initial feedback is the ranking of a list of desired products. Upon the selection of a desired category, the user is presented with a list of several (e.g. ten) products. This list of products is generated from two aspects. The first part is from the previous preference tracked by the system. If there are some previous preferred products saved within the database and the user wishes to include them, the system will refer to this history of preference, and come up with some products for the user. The second part is some random products retrieved by the system from the product database.

With the list of products, the user is to rank the products in accordance to his or her preference. Since the only properties of the products presented to the user are the product descriptions and price, the user has only these two attributes for reference. In this paper, the agents in the system give preference to a product if it is priced the highest. Hence, the highest priced and lowest priced products are ranked 1st and 10th respectively by the agents. In doing so, if the user ranks a product higher than the agent's ranking, then the hypothesis is that the user is affected by some unaccounted product attributes. From the ranking of products, we suggest two propositions as follows.

Proposition 1: If the user gives product X a higher ranking than what the agent gives, then product X is positively ranked. On the other hand, if the user gives product X a lower ranking than what the agent gives, then product X is negatively ranked.

For example, if the user ranked Product X as 2nd but the agent ranked Product X as 9th, then product X's rankingScore = (Ranking by agent) – (Ranking by user) = 9 – 2 = 7 (positive ranking). Every positively or negatively ranked product will have a rankingScore.

Proposition 2: If the user ranks product X higher than product Y, and the agent also ranks product X higher than product Y, the agent will be awarded one ranking fitness point.

3.2 DETECTING GENERIC ATTRIBUTES WITH FEATURE ANALYSIS

From proposition 1, positively and negatively ranked products are also known as illogical ranked products. Those positively illogically ranked products might contain some attributes affecting the user. In addition to the list of positive illogically ranked products, there may be some logically ranked products. The system does not wish to exclude these influential products that the user may like. For this reason, other than the positive illogically ranked products, the system will also include any “logically” ranked products in the top five ranking into the list of suspicious products. Figure 4 illustrates the aggregation.

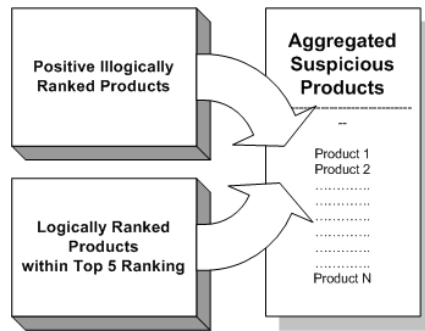


Figure 4: Aggregation of Illogically and Logically Ranked Products

The aggregated list of suspicious products is transferred to the pool of agents for evaluation. The descriptive field for each product is analyzed for the understanding of the hidden attributes in the product. Figure 5 illustrates the tokenization process. As a matter of fact, any attribute that is longer than one word will be treated as a 2-word attribute. The composite attribute will be confirmed when it is checked with the current product attribute ontology. For example, the ontology contains a product attribute “Western Digital”, which is the brand of a hard disk. If “Western” is detected in the product description, the agent will check if “Digital” appears after “Western” in

the description. If “Western” and “Digital” exist in the correct sequence, the agent will treat “Western Digital” as a single attribute instead of two separate attributes.

With the aggregated list of suspicious product attributes, the analysis of these attributes follows. In the IBM research report [20], Neumann *et al.* made use of the properties (features) of the values stored for an attribute, to determine its signature. Accurate classification was possible for their work, as their aim was to group similar attributes among various groups of data values. Contrary to our research data, each attribute in their research contains more corresponding values. As such, the IBM researchers were able to make use of an attribute signature vector storing the average number of occurrences (as a fraction) of the Boolean features for all its values.

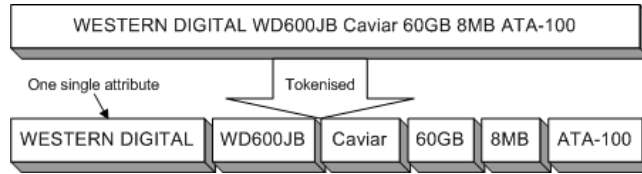


Figure 5: Tokenization of Product Descriptions

In our design, we try to cluster a new attribute into a group of attributes. For each attribute, features will be examined and noted in a Boolean feature vector whether the attribute has the feature or not. If a feature exists, its presence will be represented by “1”, otherwise “0”. Features include the existence of certain characters, as such hyphen or digits. Aggregated features, such as the presence of upper-case characters or alphanumeric words will also be examined. Features are combined to form a Boolean data signature vector describing multiple properties of a single attribute.

Having a unique signature for each product attribute is not a strong property for similarity classification, simply because a binary vector of “1” and “0” has only a 1-dimensional similarity confidence. We argue that by comparing simply the presence of a feature in an attribute, it is not strong enough to cluster similar attributes. Some slight difference in the features of attributes may occur in two similar attributes belonging to the same attribute group. For example, although “450Mhz”, “900MHZ” and “1.5Ghz” have slight disparity in their features, they are common representations of speed (generic attribute) in computer terms. Therefore we propose a more general 2-dimensional similarity confidence measurement among attributes. Besides using a binary vector of the attribute, we also apply different weights to the features, reflecting different significance. With the inclusion of feature weight, we can generate an attribute signature by aggregation of weighted data signature. However, finding a suitable weight for each feature in the attribute is a challenge by itself.

Before a feature set, which is a list of attribute features, can be formed, we need to formally define attribute feature. A feature is a unique property of an attribute. It may check for the presence of a symbol, whether the attribute contains any alphabet or digit. For the purpose of classification, a carefully selected set, called a feature set, is used as the fundamental component for generating a data signature for each data. A few definitions are as follows.

Definition 1 (Feature): A feature f is a Boolean function that takes a data value t as input and generates 1 (for true) and 0 (for false) as output.

Definition 2 (Data Signature): Given a data value t and a feature set $F = \{f_1, f_2, \dots, f_n\}$, the data signature of t with respect to F is $F(t) := (f_1(t), f_2(t), \dots, f_n(t))$.

However, as mentioned earlier, a data signature with just the feature set as the sole validation source will not provide a strong case. In this research, we propose the inclusion of weight for each feature. Weighted feature will reflect different significance. A feature weight set is a vector of feature weights. Therefore, an attribute together with its feature set and feature weight set will generate a unique attribute signature.

Definition 3 (Feature Weight Set): Given a data value t , a feature set F and n number of features, the feature weight set $Wt(t) := (w_1, w_2, \dots, w_n)$, where $0 \leq w \leq 1.0$.

Definition 4 (Attribute Signature): Given an attribute a with n existing features, f_1, f_2, \dots, f_n , a feature set F and a feature weight set Wt , the attribute signature of a is

$$\delta(a) := \sum_{i=1}^n f_i(a) * w_i \quad (1)$$

With the above definitions, we can determine similarity or dissimilarity in attributes. For example, assume attribute data $a_1 = 500\text{mhz}$, $a_2 = 750\text{MHZ}$, $a_3 = \text{Maxtor}$ and $a_4 = 1.2\text{Ghz}$. On top of that, let f_1 check whether the input contains any digit, f_2 check whether the input contains any upper-case letters, f_3 check whether the input contains any lower-case letter, and f_4 check for numeric-alpha pattern. We have the following.

Feature Set $F = \{f_1, f_2, f_3, f_4\}$

Data signature $F(a_1) = (1, 0, 1, 1)$

$$F(a_2) = (1, 1, 0, 1)$$

$$F(a_3) = (0, 1, 1, 0)$$

$$F(a_4) = (1, 1, 1, 1)$$

Assume the feature weight set $W_t = \{ 0.65, 0.2, 0.2, 0.85 \}$. Therefore with a_1, a_2, a_3 and a_4 defined earlier, we generate the following attribute signatures by using (1),

$$\delta(a_1) = (1*0.65) + (0*0.2) + (1*0.2) + (1*0.85) = 1.70$$

$$\delta(a_2) = (1*0.65) + (1*0.2) + (0*0.2) + (1*0.85) = 1.70$$

$$\delta(a_3) = (0*0.65) + (1*0.2) + (1*0.2) + (0*0.85) = 0.40$$

$$\delta(a_4) = (1*0.65) + (1*0.2) + (1*0.2) + (1*0.85) = 1.90$$

From this simple example, with only 4 features and weights, the proximity of attribute signatures of a_1, a_2 and a_4 suggest similarity among them, and dissimilarity in attribute a_3 . In fact, attribute a_3 is a brand name while the other attributes are speeds in computer terms.

On a closer look at the above derivation of attribute signature, it can be seen that an attribute signature is just the sum of all existing weighted features (represented by “1”). The approach was inspired by the hamming distance function in the communication field. In our case, we made some variation where weight is added to reflect significance.

There are two types of features that are used in this paper – singleton and aggregate features. A singleton feature checks for the presence of certain characters or symbols in the attribute data. On the other hand, an aggregate feature checks if the attribute data contains any characters in a predefined aggregate set of characters. An aggregate

set aims to capture characters that are strongly correlated at various levels. Observing the satisfactory results from Neumann *et al.* research [20], this research uses both singleton and aggregate features. The combination usage of features has shown to yield the best results.

In this research, all singleton features check for a delimiter, a letter, or a digit. For aggregate features, delimiters, lower-case vowels, upper-case vowels, lower-case letters, upper-case letters, alphabets, digits, alphanumeric and numeric-alpha are verified during evaluation. Table 1 tabulates the list of features.

Table 1: The Default Feature Set

Singleton Feature	
A Digit	{0}, ..., {9}
A Letter	{a}, ..., {z}, {A}, ..., {Z}
A Delimiter	{@}, ..., {\}
A Lower Vowel	{a}, {e}, {i}, {o}, {u}
A Upper Vowel	{A}, {E}, {I}, {O}, {U}
A Lower Letter	{a}, ..., {z}
A Upper Letter	{A}, ..., {Z}
Hyphen	{-}
Aggregate Feature	
Digits	{0, ..., 9}
Letters	{a, ... z}, {A, ..., Z}, {a, ..., z, A, ..., Z}
Lower Letters	{a, ... z}
Upper Letters	{A, ..., Z}
Alphanumeric	{a, ..., z, A, ..., Z, 0, ..., 9}
Numeric Alpha	{0, ..., 9, a, ..., z, A, ..., Z}

3.3 DETECTING GENERIC PREFERENCE

After the creation of attribute signatures from suspicious product attributes, the ability of the agents to classify these attribute signatures is vital. Only with efficient and accurate classification can there be a generalization of attributes that provides the system with some clues to the customer's preference. For example, 500Mhz, 750MHZ and 1.2Ghz can be clustered under a generic attribute grouping such as "Speed".

We use a classification approach based on an adaptation of the k-Nearest Neighbor (k-NN) classification technique in attributes clustering, which is simple yet competent. There are many cases [5] [21] where the k-NN approach has been deployed in webpage and text classification with good results. In particular, the k-NN classification algorithm used by Han *et al.* [5] is able to learn the importance of attributes and utilize them in the similarity measurement.

For the k-NN approach in this paper, there does not exist an explicit training procedure usually found in the traditional k-NN approach. Instead, the main training and learning process is found in the genetic-algorithmic optimization of the agents. Each new attribute to be classified will have its attribute signature verified with every existing generic attribute grouping. A proximity level has to be achieved in order for that new attribute to be clustered into the generic attribute category.

Figure 6 illustrates the k-NN classification approach. There are three existing groups: Neighborhood A, B and C, with a, b and c being the number of instances respectively.

The average value of each neighborhood is calculated using (2), where there are k instances of attributes in that neighborhood h and δ is the attribute signature.

$$\text{Average value of neighborhood } \mu(h) := \frac{\sum_{i=1}^k \delta(i)}{k} \quad (2)$$

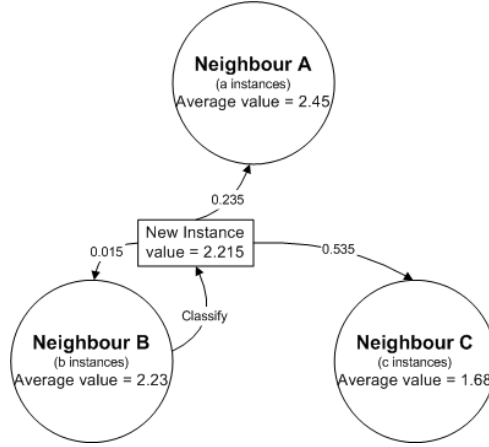


Figure 6: Nearest Neighbor Approach

For example, a new instance of value (attribute signature value) 2.215 is introduced. Upon verification with all three groupings, the new instance best matches the average value in neighborhood B, because it has the lowest proximity value of 0.015 with neighborhood B. Therefore the new instance is classified under neighborhood B. Proximity value is calculated as the absolute Euclidean distance between the new instance and the average value of a neighborhood, as shown in (3).

$$\text{Proximity value of instance } s: \rho(s) := |\delta(s) - \mu(h)| \quad (3)$$

With distinctive attribute signature, the agent is able to classify similar attributes into a generic attribute class. However, a small problem is that the agent does not know the name of that generic classification. For example, the agent might know that “500Mhz”, “850MHZ” and “1.5Ghz” belong to the same generic classification, but it

does not know the name for this classification without prior knowledge. Therefore, to aid the agent in the discovery of the name for any generic classes, a preloaded product attribute ontology is referred to. Generic classes are defined as generalized groupings of similar attributes in this paper.

Product attribute ontology contains information about the relationships and schematics between attributes and product, as shown in Figure 2. In this research, it is assumed that there is some prior knowledge about some default product attribute ontology in the database. It will serve as a basic initial reference for the agents. The ontology is available to the population of agents when the user chooses the category of a product.

3.4 GA-BASED EVOLUTION AND FEATURE WEIGHT OPTIMIZATION

A good feature weight set enables the generation of distinctive attribute signatures, which aids in good classification of attributes. In this paper, a Genetic Algorithm (GA) based optimization approach is used to find the suitable feature weight set. Feature weight ranges from 0.0 to 1.0 in a double precision format. The chromosomes are manipulated in binary string format with eight bits. Thus, 0.0 will be coded as 00000000 and 1.0 coded as 11111111. All the coded binary strings are concatenated to form a long binary string representing the feature weights set.

The GA design in this paper uses a modest form of elitist selection, in which a percentage of elite agents from the previous generation replaces the same percentage of poorly performing agents in the current generation. Elitism rapidly increases the

performance of GA, by preventing loss of the best-found solutions. The GA begins with the selection process, where a tournament-like selection operator is used. Three randomly selected chromosomes participate in a round of tournament, where the fittest of the three will be selected as one of the parents for mating. During each round of selection, the three random chromosomes selected during the 1st round of tournament cannot enter the 2nd round of selection for the 2nd parent. This blockade of selection is to ensure diversity in mating. Figure 7 illustrates the selection process.

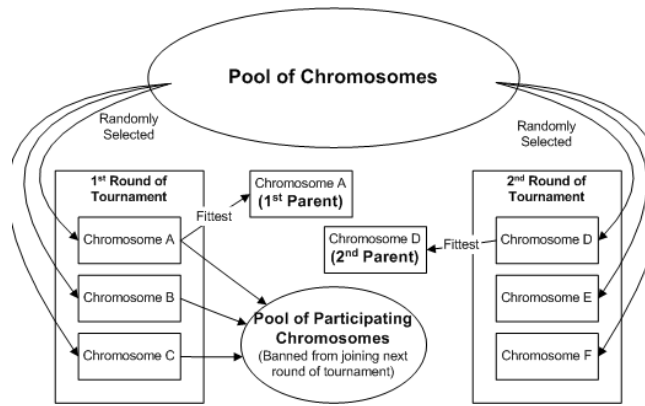


Figure 7: Tournament Selection Process

After the tournament selection, the two parents will undergo a crossover operation. A randomized single-point crossover is performed on these two parents to produce two offspring. After the crossover operation, the chromosomes undergo a mutation. Mutation rate is the probability that a bit in the chromosome will undergo a mutation. Since the chromosome is in the binary string format, a mutated bit will have its value flipped, such that “0” becomes “1” and “1” becomes “0”.

The fitness of chromosome is evaluated through the agent fitness. The agent fitness function comprises two components. The first component is the ranking fitness and the second is the evolution fitness.

$$\text{Agent Fitness} = \text{Ranking Fitness} + \text{Evolution Fitness} \quad (4)$$

From proposition 2, the agent is awarded some ranking fitness point when the user and the agent both rank product X higher than product Y. Furthermore, if the agent ranks the 1st ranking product in the same order as the user, the agent can receive up to a maximum of 9 ranking fitness points. Extending similar treatment to the 2nd ranking product, the agent can receive up to 8 ranking fitness points for that 2nd product ranking. If we extend similar treatment to all 10 products on the list, and considering that the agent has ranked all products in the same order as the user, then the maximum total ranking fitness points that the agent can receive is as follows.

$$\text{Total Ranking Fitness} = 9 + 8 + \dots + 2 + 1 = 45 \quad (5)$$

On the other hand, agents are awarded points during evolution for good classification of attributes and have points deducted for poor classification. The points that an agent receives during evolution will be known as evolution fitness. The system will reward one evolution fitness point to an agent for the detection of a generic classification of attributes. For example, if the agent derives 3 generic classes after the classification process, it will be rewarded 3 fitness points. However, for the agents to learn correctly, we also give demerit points if any agent has made some wrong or inaccurate classification. The user is allowed to view all detected classifications by the elite agents, and is able to remove or edit the classification. Any removal or amendment of the classification proposed by the agent will result in demerit points. A removal will

penalize more than an amendment, because a removal signifies a complete rejection by the user.

```
For all GENERATIONS {
  For half the POPULATION size {
    parentA <= Select from Population_A based on evaluation
    parentB <= Select from Population_A based on evaluation

    child1, child2 <= Generate 2 children based on
                        crossover(ParentA, Parent B)

    child1 <= Mutate(child1, MUTATION RATE)
    child2 <= Mutate(child2, MUTATION RATE)

    Population_B <= Add child1 to Population_B;
    Population_B <= Add child2 to Population_B;
  }

  Evaluation_B <= evaluate (Population_B)
  Elites replace poorly performed agents in Population_B
  Population_A <= new Generation(Population_B)
  Evaluation_A <= evaluate (Population_A)
  Elites <= find elite agents from Evaluation_A
}
```

Figure 8: Genetic Algorithm

After all the manipulations of the agents' feature weight set, the current agent population must first be evaluated before the evolution process. The evaluation will determine the initial elite agents. After retrieving the elite agents, the population of agents will loop through a number of generations. During each generation, 2 parents are selected based on the tournament selection approach mentioned earlier. Figure 8 shows the GA pseudo code. The new generation, also known as Population B is evaluated, and those poor performers in terms of agent fitness are to be replaced by elite parents from the previous generation, also known as Population A. The number of elite parents is determined by the elitist rate defined. Elitism allows a certain number of parents which are very fit to be carried over to the new generation untouched. The new population with elitist replacement then undergoes the next

round of evolution process consisting of selection, recombination, mutation and evaluation.

The intelligent agents in the system have learning capabilities through incremental re-evaluation. The incremental detection approach undertaken in understanding user's preference will enable the system to analyze complex user preference situation. The system recognizes that not all generic attributes can be detected accurately within one feedback cycle. Therefore, the system seeks to consider any previous results. The satisfied recommendations and generic attributes affecting a user's preference in one feedback will become useful information in the next set of feedback. Reusing vital information previously found while new set of generic attributes and products are being detected allows the software agents to learn incrementally. However, the user has the choice to reuse previously detected information or to start afresh in each feedback cycle.

4. EVALUATION OF DESIGN

For the results shown in the following subsections, some default simulation parameters are used. During the GA evolution, a population of 30 agents is evolved. During the selection process of GA, tournament selection with a tournament size of 3 is implemented. The elitist rate is set at 0.1, or 10%, which has been found to be suitable and appropriate. In addition, the mutation rate is set at 0.25, which gives better results than other mutation rates during the simulation.

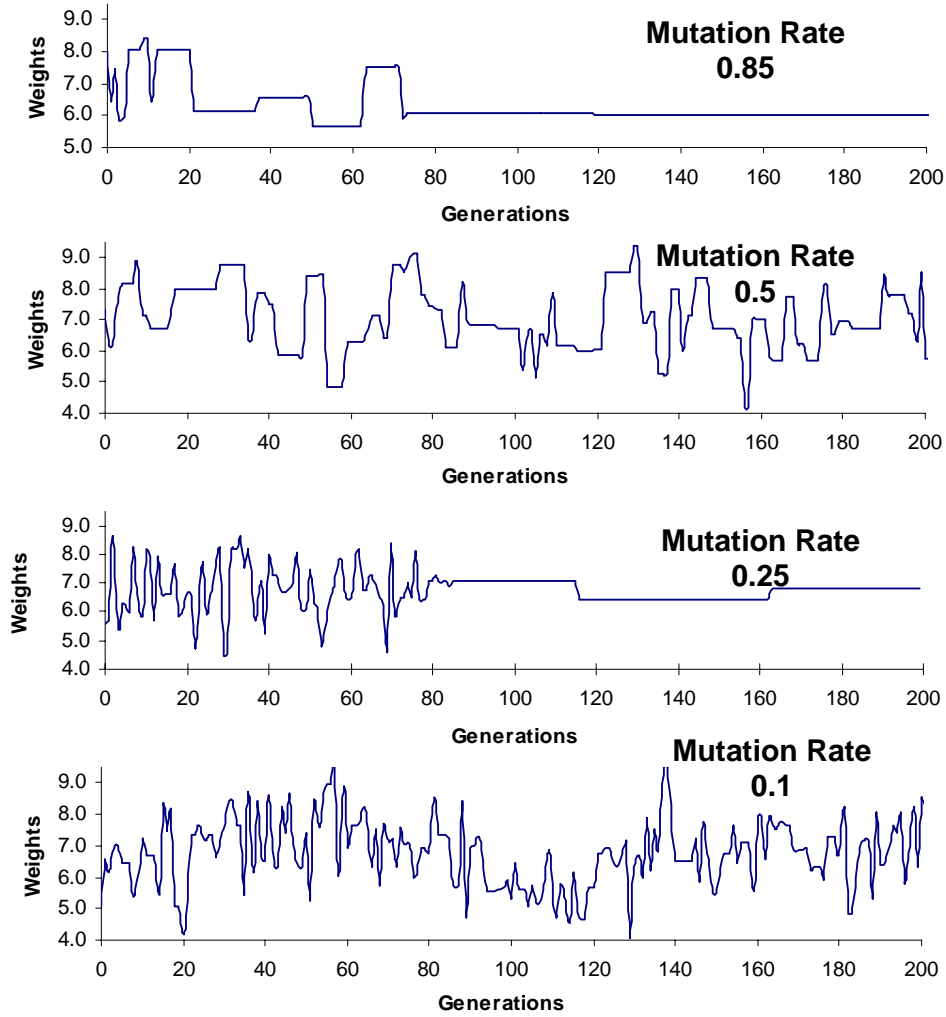


Figure 9: Convergence of Feature Weight for Four Mutation Rates

4.1 Convergence of Feature Set Weights

To evaluate if our prototype can obtain a suitable set of feature weight, we run a simulation with some worst-case illogical ranking initially. Worst-case illogical ranking occurs when the user ranks the cheapest product top. HardDisk is chosen as the product category for the simulation. Four different mutation rates were used which is illustrated in Figure 9. As observed, a mutation rate at 0.1 or 0.5 did not result in the

convergence of feature weights. On the other hand, a mutation rate at 0.25 or 0.85 resulted in the convergence of feature weights. However, it should be noted that the convergence for mutation rate at 0.25 is preferred because there is a greater variation of weights before convergence. The variation allows the agents to learn different sets of weights before convergence, which is more desirable than in the case of mutation rate of 0.85. When the feature set weights have converged and stabilized, the agents would have found a set of suitable feature weight for each attribute that may result in good classification of generic attributes.

4.2 Misclassification Rate

Using a mutation rate at 0.25 and single crossover, a simulation was conducted with 12 evolutions. Each evolution process consists of 30 agents with a set of user feedback regarding the classification of attributes. The misclassification rate is defined as the percentage of classes or attributes that are classified wrongly.

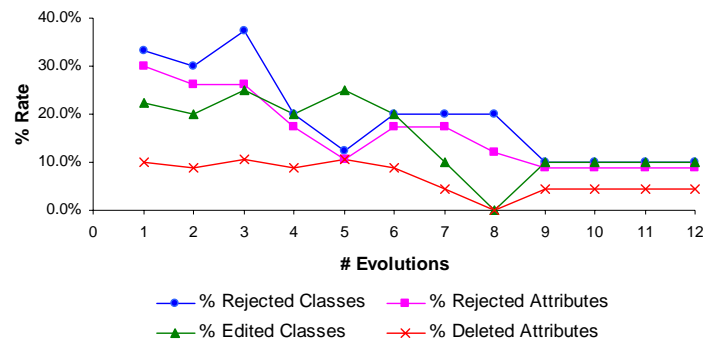


Figure 10: Misclassification Rate

Figure 10 illustrates the misclassification rates of rejected classes, edited classes, rejected attributes and deleted attributes. These feedbacks are from the user who will inspect the classification produced by the agents after every evolution. From Figure

10, the misclassification rates are noted to be decreasing with more evolutions. After 9 evolutions, the average misclassification rates drop below 10%. From the 11th to the 12th evolution, there is no improvement in classification. This shows that the GA-agents are learning well through the process of evolutions, but the learning rate reaches a threshold after 10 evolutions when the misclassification drops just below 10%. The misclassification rate of 10% cannot be further improved in this case, which could be due to the fact that there are a limited number of features (e.g. 14) at the disposal of the agents for the generation of attribute signature. The limitation in the variety of attribute features reduces the accuracy of classification of similar attributes. A more distinctive and unique feature set will be needed for more accurate classification. By and large, a 10% misclassification rate would be very satisfactory considering the complexity of classifying different types of attributes from numerical, textual to alphanumeric.

4.3 Detecting Generic Attributes

This section will discuss the ability of the prototype to detect generic attributes and its ability to adapt to user's abrupt changes.

Rank	Category	Description	Price (S\$)
1	HardDisk	WESTERN DIGITAL CaviarAB 100GB ATA-100	223
2	HardDisk	WESTERN DIGITAL WD600BB Caviar 80GB 2MB ATA-100	170
3	HardDisk	WESTERN DIGITAL WVD600JB Caviar 60GB 8MB ATA-100	169
4	HardDisk	SEAGATE 80GB Barracuda ATA-100	186
5	HardDisk	SEAGATE 40GB U Series 40810ST340810A	120
6	HardDisk	SEAGATE 40GB Barracuda ATA-100	136
7	HardDisk	SEAGATE U Series 5 30.0GB ATA-100	148
8	HardDisk	QUANTUM FireballPlus AS 60GB ATA-100	243
9	HardDisk	QUANTUM FireballPlus AS 30GB ATA-100	169
10	HardDisk	MAXTOR DiamondMax D541X 10GB 4D040H2	89

Figure 11: Initial Product Ranking

In this test case, the user wishes to search for a hard disk and considers the brand of hard disk as an important attribute influencing his preference. In particular, the user

has a preference for WESTERN DIGITAL and SEAGATE hard disks as indicated in Figure 11, where WESTERN DIGITAL and SEAGATE hard disks are ranked top.

The ranked products are submitted for evaluation by the population of agents. In total 30 autonomous agents evaluate the input by the user. When the agents have finished all the evolution processes, the elite agents will provide some generic classes detected, as shown in Figure 12. As observed, the system has successfully detected some generic classes such as the “Brand” and “Size”. The user can reject any undesirable classification by de-selecting the generic classes. The names of the generic classes that match the names found in the ontology as shown in Figure 13 are available for the user to confirm.

Index	Generic Group	Attributes	Proximity	Agent#	Confirm
1	Brand	WESTERN DIGITAL, SEAGATE	2.11	0	<input checked="" type="checkbox"/>
2	Generic#2	CaviarAB, 40810ST340810A	4.065	0	<input checked="" type="checkbox"/>
3	Size	100GB, 80GB, 2MB, 60GB, 8MB, 40GB	2.94	0	<input checked="" type="checkbox"/>
4	Generic#4	ATA-100	4.2	0	<input checked="" type="checkbox"/>
5	Generic#5	WD800BB, WD600JB	3.76	0	<input checked="" type="checkbox"/>
6	Generic#6	Caviar, Series, Barracuda	3.1	0	<input checked="" type="checkbox"/>

Figure 12: Detected Generic Classes

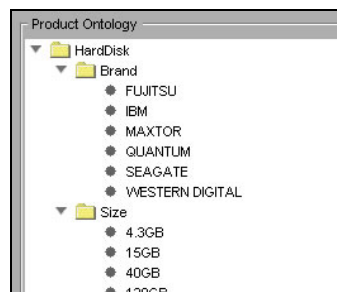


Figure 13: HardDisk Attribute Ontology

When the user feedbacks to the agents about the generic classes and re-evaluate the attributes again, the agents are able to come up with some recommended products as shown in Figure 14.

Index	Category	Description	Price (\$S)
1	HardDisk	WESTERN DIGITAL CaviarAB 100GB ATA-100	223
2	HardDisk	WESTERN DIGITAL WD800BB Caviar 80GB 2MB ATA-100	170
3	HardDisk	WESTERN DIGITAL WD600JB Caviar 60GB 8MB ATA-100	169
4	HardDisk	SEAGATE 40GB U Series 40810ST340810A	120
5	HardDisk	SEAGATE 40GB Barracuda ATA-100	136

Save Results

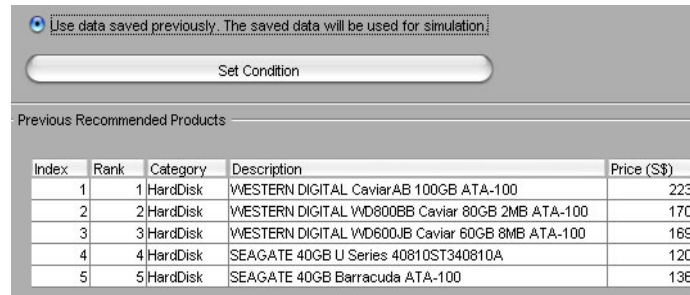
Figure 14: Recommended Products by Elite Agents

The recommendation indeed provides products that the user preferred, such as WESTERN DIGITAL and SEAGATE hard disks. The recommendation relies mainly on the relationship between successful detection of generic attributes and positive ranking of products by the user. Therefore, only products that are highly ranked yet containing the generic attributes would be recommended. Because the generic attributes play an important role in influencing the user to rank the products highly, we deduce that the acknowledged generic attributes that can be found in the highly ranked products are usually the generic preference of the user.

Now that the user has saved the 1st round of information, he can go for a 2nd round and makes use of the previously saved data. Figure 15 shows the previously saved data and the user shall set the initial condition with these data. Making use of previously saved information allows the agents to incrementally learn about the user preference. During the 2nd round of feedback, the user ranks with WESTERN DIGITAL as the preferred choice of hard disk. The ranking is illustrated in Figure 16.

In this round, a 2nd agent also detected a generic class of brands, comprising WESTERN DIGITAL and SEAGATE. With the rejection and amendment done, the

agents will re-evaluate the product attributes and provide a second round of generic attributes list as shown in Figure 17. We can observe that the detection of generic attributes has improved with more accurate classification. The classes BRAND and SIZE are confirmed by the user.



Use data saved previously. The saved data will be used for simulation.

Set Condition

Previous Recommended Products

Index	Rank	Category	Description	Price (\$)
1	1	HardDisk	WESTERN DIGITAL CaviarAB 100GB ATA-100	223
2	2	HardDisk	WESTERN DIGITAL WD800BB Caviar 80GB 2MB ATA-100	170
3	3	HardDisk	WESTERN DIGITAL WD600JB Caviar 60GB 8MB ATA-100	169
4	4	HardDisk	SEAGATE 40GB U Series 40810ST340810A	120
5	5	HardDisk	SEAGATE 40GB Barracuda ATA-100	136

Figure 15: Making Use of Previously Saved Data

Rank	Category	Description	Price (\$)
1	HardDisk	WESTERN DIGITAL WD800BB Caviar 80GB 2MB ATA-100	170
2	HardDisk	WESTERN DIGITAL WD600JB Caviar 60GB 8MB ATA-100	169
3	HardDisk	WESTERN DIGITAL WD600BB Caviar 60GB 2MB ATA-100	163
4	HardDisk	WESTERN DIGITAL CaviarAB 100GB ATA-100	223
5	HardDisk	SEAGATE 40GB U Series 40810ST340810A	120
6	HardDisk	SEAGATE 40GB Barracuda ATA-100	136
7	HardDisk	QUANTUM FireballPlus AS 60GB ATA-100	243
8	HardDisk	MAXTOR D540X 40GB 4D040H2/4K040H2 ATA-100	119
9	HardDisk	MAXTOR D540DX 20GB4D020H1/4K020H1 ATA-100	105
10	HardDisk	MAXTOR 536DX 100GB 4W100H6 ATA-100	90

Figure 16: 2nd Round Product Ranking

Figure 18 shows the recommended products in the 2nd round. Again three WESTERN DIGITAL hard disks are recommended, but this time with only one SEAGATE hard disk. The system has probably understood that the user might not have much preference for SEAGATE as compared to WESTERN DIGITAL. In addition, though during the ranking of products, the user chooses four WESTERN DIGITAL hard disks, but only three are recommended. The fourth WESTERN DIGITAL hard disk is ranked 4th even though it is priced higher than the other three WESTERN DIGITAL hard disks. Therefore, the system detects that this WESTERN DIGITAL hard disk is

negatively ranked. In our case, negatively ranked products are not recommended to the user.

Generic Classes Detected					
Index	Generic Group	Attributes	Proximity	Agent#	Confirm
1	Brand	WESTERN DIGITAL, SEAGATE	2.52	0	<input checked="" type="checkbox"/>
2	Generic#2	WD800BB, WD600JB, WD600BB	5.02	0	<input type="checkbox"/>
3	Generic#3	Caviar, Series	2.44	0	<input type="checkbox"/>
4	Size	80GB, 2MB, 60GB, 8MB, 40GB	4.42	0	<input checked="" type="checkbox"/>
5	Generic#5	ATA-100	4.11	0	<input type="checkbox"/>

Figure 17: Re-evaluated Detected Generic Attributes

Recommended Products			
Index	Category	Description	Price (\$\$)
1	HardDisk	WESTERN DIGITAL WD800BB Caviar 80GB 2MB ATA-100	170
2	HardDisk	WESTERN DIGITAL WD600JB Caviar 60GB 8MB ATA-100	169
3	HardDisk	WESTERN DIGITAL WD600BB Caviar 60GB 2MB ATA-100	163
4	HardDisk	SEAGATE 40GB U Series 40810ST340810A	120

Figure 18: Recommended Products in the 2nd Round

We further test the adaptability of the prototype to abrupt user preference changes. In this 3rd round of feedback, the user changes his preference for quality product which is in correlation to the pricing. Therefore the user ranks the products in accordance to their prices as illustrated in Figure 19. After a few rounds of evaluations, the agents finally recommend a list of hard disks as shown in Figure 20.

Rank	Category	Description	Price (\$\$)
1	HardDisk	MAXTOR DiamondMax D540X 120GB 4G120J6 ATA-133	295
2	HardDisk	WESTERN DIGITAL CaviarAB 100GB ATA-100	223
3	HardDisk	IBM 180GXP 60GB ATA-100 FDB 2MB	195
4	HardDisk	WESTERN DIGITAL WD800JB Caviar 80GB 8MB ATA-100	187
5	HardDisk	WESTERN DIGITAL WD800BB Caviar 80GB 2MB ATA-100	170
6	HardDisk	WESTERN DIGITAL WD600JB Caviar 60GB 8MB ATA-100	169
7	HardDisk	SEAGATE 40GB Barracuda ATA-100	136
8	HardDisk	SEAGATE 40GB U Series 40810ST340810A	120
9	HardDisk	MAXTOR 536DX 40GB 4V040H3 ATA-100	119
10	HardDisk	MAXTOR D540DX 20GB4D020H1/4K020H1 ATA-100	105

Figure 19: 3rd Round Product Ranking

Recommended Products			
Index	Category	Description	Price (\$\$)
1	HardDisk	MAXTOR DiamondMax D540X 120GB 4G120J6 ATA-133	295
2	HardDisk	WESTERN DIGITAL CaviarAB 100GB ATA-100	223
3	HardDisk	WESTERN DIGITAL VWD800JB Caviar 80GB 8MB ATA-100	187
4	HardDisk	WESTERN DIGITAL VWD800BB Caviar 80GB 2MB ATA-100	170
5	HardDisk	IBM 180GXP 60GB ATA-100 FDB 2MB	195

Figure 20: Recommended Products in the 3rd Round

From the recommendation shown in Figure 20, we observed that system is able to adapt to the change of preference from BRAND to PRICE. Maxtor DiamondMax is the top recommended product with a price of \$295.00. Though most of the recommended products are highly priced quality hard disks, it can be noted that the \$195.00 IBM hard disk is placed below the WESTERN DIGITAL hard disk. The WESTERN DIGITAL hard disks though priced lower than the IBM hard disk is recommended above it. This shows that the system has adapted to the change in user preference, nevertheless it also takes into account the generic attributes detected in the previous rounds.

To show that the system has indeed adapted to the changes in user preference, we tracked the feature weights during each generation of evolutions. Figure 21 shows the continuous tracking of feature weights from the 1st to 450th generation of evolutions by the agents. The 450 generations comprise the 3 rounds of simulations in chronological order. It is observed that the weights stabilized within the 1st round of simulations which end at generation 200th. In the 2nd round, from generation 200th to generation 325th, there is some fluctuation in the weights initially. However the weights are stabilized when it reaches the 300th generation.

During the 1st and 2nd rounds, the user has a preference for BRAND, especially WESTERN DIGITAL and SEAGATE. The 3rd round begins at generation 325th and we can see that there is a sudden fluctuation in the feature weights. This is likely due to the fact that in the 3rd round, the user has a sudden preference for PRICE, and disregarding the preference for BRAND. The system takes around two hundred generations before the feature weights stabilized. The system has tried and successfully adapted to the sudden change in user preference, with a new set of feature weights.

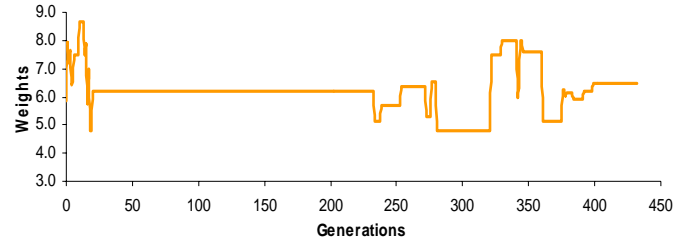


Figure 21: Adapting to User Preference Changes

4.4 Summary and Discussions

On the whole, the recommendation provided based on the detection of generic attributes is found to be satisfactory. Most recommendations provide products tailored to the user's preference. However, with more rounds of re-evaluations and feedbacks, the agents are able to improve the recommendation quality. The complexity of feature weights with numerous features within each attribute is closely related to the system performance. However, GA is able to overcome this complexity and produce satisfactory overall population fitness over generations. The misclassification rates of attributes also decreased over generations with final misclassification rates at around

10%, which produced satisfactory results in our case. It is also observed that the system is able to adapt to abrupt user-preference changes.

Many other optimization approaches exist that may work as well for our research problem. In particular, Simulated Annealing (SA) and Tabu Search (TS) techniques are both comparable to the GA approach. As observed in [22] [23], SA is comparable to GA, as SA can be thought of as GA with a population size of one. Empirically, SA is able to achieve satisfactory results in shorter time, but will not improve much when given more time to progress. In contrast, though GA may be a slower starter, it has the ability to improve the solution consistently over time, or even attain more and better solutions. In fact, we are able to complete the optimization process in this research using GA in a relatively short time span. TS has been proven successful in target-specific problems with the best solution in mind. However, the problem of optimizing multiple-attribute fitness in our research cannot be classified as a target-specific problem because our agents cannot predetermine exactly the user's preference, since a "good" attribute may only have temporal wellness. Besides, TS concentrates on navigating towards maxima or minima fitness and avoids repetition on a tested path. Such characteristic of TS will pose a problem when the user's preference changes.

In this paper, GA is chosen for the optimization problem partly because GA is highly adaptive. A highly adaptive algorithm is needed for two reasons. Firstly, the user's taste might not be consistent, so the agents have to be flexible in their findings to adapt to any changes. Secondly, the algorithm might make a mistake during optimization, and the nature of GA allows the agents to revisit some of the solutions

which they have discarded previously. The solutions presented in this research require an optimization process to be self-correcting in nature, and GA's adaptive capability will serve this research well. According to Chu and Fang [24], GA can produce numerous, diverse near-optimal solutions simultaneously because GA holds the whole generation of chromosomes which may not originate from the same parents. In fact, GA also exhibits the capability of parallelism by searching solutions from many points in the search space, rather than from just one starting point [25].

5. CONCLUSION

This paper proposed a new approach to detect user's generic preference with feature selection. The product attributes were evaluated according to their distinctive features. Attribute signature and weight set were proposed to detect the generic attributes. GA was used to aid in the optimization of feature weights. A prototype was implemented and had shown that the detection of generic attributes was feasible with correct attribute classification and some help from the product attribute ontology. It was also observed that the system was able to adapt to abrupt customer-preference changes.

For future research, different types of features are potential candidates for successful classification such as schema meta-data, n-grams, entire words, and domain specific features. Furthermore, the current system relies mainly on the user and product attribute ontology to provide accurate generic attribute name for any classification. However, the ontology implemented is simple and without much intelligence. Further development in including better ontology with product-attribute relationships and associations will help the system detect generic attributes better.

REFERENCES

- [1] H.S. Nwana, D. Ndumu, An introduction to agent technology, *BT Technology Journal* 14 (4) (1996).
- [2] H.H. Sung, Helping online customers decide through web personalization, *IEEE Intelligent Systems*, 17 (6) (2002) 34-43.
- [3] B. Sheth, P. Maes, Evolving agents for personalized information filtering, in: *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*, 1993, pp. 345-352.
- [4] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, *IEEE Internet Computing*, 7 (1) 2003) 76-80.
- [5] E.H. Han, G. Karypis, V. Kumar, Text categorization using weight adjusted k-nearest neighbor classification, in: *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2001, pp. 53-65.
- [6] S. Luke, L. Spector, D. Rager, J. Hendler, Ontology-based web agent, in: *Proceedings of the 1st International ACM Conference on Autonomous Agents*, 1997, pp. 59-66,
- [7] S.U. Guan, F. Zhu, Ontology acquisition and exchange of evolutionary product-brokering agents, *Journal of Research and Practice in Information Technology* 36 (1) (2004) 35-46.
- [8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1996.
- [9] S.U. Guan, F. Zhu, Incremental learning of collaborative classifier agents with new class acquisition - an incremental genetic algorithm approach, *International Journal of Intelligent Systems* 18 (11) (2003) 1173-1193.
- [10] S.U. Guan, C.S. Ngoo, F. Zhu, Handybroker - an intelligent product-brokering agent for m-commerce applications with user preference tracking, *Electronic Commerce and Research Applications*, 1 (3-4) (2002) 314-330.
- [11] H. Shibata, T. Hoshiai, M. Kubota, M. Teramoto, Agent technology recommending personalized information and its evaluation, in: *Proceedings of the 2nd International Workshop on Autonomous Decentralized System*, 2002, pp. 176-183.
- [12] S.U. Guan, F. Zhu, Agent fabrication and its implementation for agent-based electronic commerce, *International Journal of Information Technology and Decision Making* 1 (3) (2002) 473-489.
- [13] S.U. Guan, F. Zhu, M.T. Maung, A factory-based approach to support e-commerce agent fabrication, *Electronic Commerce and Research Applications* 3 (1) (2004) 39-53.
- [14] S.J. Jung, J.H. Hong, T.S. Kim, A formal model for user preference, in: *Proceedings of the IEEE International Conference on Data Mining*, 2002, pp. 235-242.
- [15] Y.T. Guo, J.P. Müller, C. Weinhardt, Elicitation of user preferences for multi-attribute negotiation, in: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne Australia, 2003, pp. 1004-1005.
- [16] S.U. Guan, P.C. Tan, T.K. Chan, Intelligent product brokering for e-commerce: an incremental approach to unaccounted attribute detection, *Electronic Commerce Research and Applications* 3 (3) (2004) 232-252.
- [17] C. Shahabi, Y.S. Chen, An adaptive recommendation system without explicit acquisition of use relevance feedback, *Source Distributed and Parallel Databases*, 14 (2) (2003) 173-192.

- [18] M. Kwak, D.S. Cho, Collaborative filtering with automatic rating for recommendation, in: Proceedings of the International Symposium on Industrial Electronics, 2001, pp. 625-628.
- [19] V. Tamma, M. Wooldridge, I. Blacoe, I. Dickinson, An ontology based approach to automated negotiation, in: J. Padget et al. (Eds.): Agent-Mediated Electronic Commerce IV: Designing Mechanisms and Systems, LNAI 2531, Springer-Verlag, Berlin Heidelberg, 2002, pp. 219–237.
- [20] F. Neumann, C.T. Ho, X. Tian, L. Haas, N. Meggido, Attribute classification using feature analysis, in: Proceedings of the International Conference on Data Engineering ICDE, 2002.
- [21] O.W. Kwon, J.H. Lee, Web page classification based on k-nearest neighbor approach, in: Proceedings of the Fifth International Workshop on Information Retrieval with Asian languages, Hong Kong, 2000, pp. 9-15.
- [22] L. Davis, (Eds.) Genetic Algorithms and Simulated Annealing. Research Notes in Artificial Intelligence, Pitman Publishing, 1987.
- [23] L. Ingber, B. Rosen, Genetic algorithms and very fast simulated reannealing: a comparison, Mathematical Computer Modeling 16 (11) (1992) 87-100.
- [24] S.C. Chu, H.L. Fang, Genetic algorithms vs. tabu search in timetable scheduling, in: Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, 1999, pp. 492-495.
- [25] E. Alba, M. Tomassini, Parallelism and evolutionary algorithms, IEEE Transactions on Evolutionary Computation 6 (5) (2002) 443-462.